Mocks

ЗАГЛУШКИ

Mock-объект (от англ. mock object, букв. — «объект-пародия», «объект-имитация», а также «подставка») — в объектно-ориентированном программировании — тип объектов, реализующих заданные аспекты моделируемого программного окружения.

Заглушка — это небольшая часть кода, которая заменяет собой другой компонент во время тестирования. Преимущество использования заглушки заключается в том, что она возвращает последовательные результаты, упрощая написание теста. Тесты можно выполнять, даже если другие компоненты пока не работают.

Современные приложения редко работают в изоляции, чаще всего они интегрируются с множеством сторонних сервисов. Поэтому остро встает вопрос тестирования этой интеграции в рамках интеграционного или функционального тестирования. В процессе, если тестировать напрямую, то возникает ряд проблем:

Тесты замедляются достаточно сильно. Доступ к внешнему сервису обычно осуществляется по сети с использованием публичных протоколов, что гораздо медленнее локальной интеграции. Внешний сервис обычно не такой быстрый и подвержен собственной нагрузке, во время которой может замедляться еще сильнее. В результате я встречал на практике замедление интеграционного

набора тестов с 10 минут до пары часов, причем время выполнения сильно плавало.

Тесты становятся менее стабильными. Внешние сервисы подвержены нагрузкам, иногда недоступны из-за обновлений, «падают» при неверном обновлении разработчиками сервиса, ну и сетевые ошибки также не исключены. Все это очень сильно влияет на ваш тестовый набор и его стабильность. Тесты «падают» неожиданно, на изучение проблемы уходит много времени и члены команды начинают недоверчиво относиться к тестам в целом. В результате сама полезность тестирования может быть поставлена под сомнение.

Вы просто не имеете возможность получить некоторые ответы внешнего сервиса (ошибки, специфические данные и т.д.), что может сказаться на покрытии вашего кода тестами, а значит надежности регрессионного тестирования интеграционными тестами. Ведь нужно покрывать не только позитивные, но и негативные сценарии.

Реальный сервис не всегда имеет тестовый интерфейс и вам приходится использовать совершенно реальный. А это не всегда бесплатно и любые ошибки в вашем коде подвергают риску реальные данные на внешнем сервисе. Благодаря этой проблеме можно легко удалить очень важные данные или заблокировать доступ из реального приложения. И это не говоря о том, что много денег может быть потрачено на «неожиданное» использование внешнего сервиса.

Заглушки необходимы при изменении состава интегрируемых модулей. Они могут заменить модули, которые вызываются тестируемым модулем. Заглушка может выполнять минимальную обработку данных, имитирует прием и возврат данных. Но использование заглушек приводит к дополнительны затратам, так как они не поставляются с конечным программным продуктом.

Существует 4 вида заглушек:

1. **Заглушка-флажок.** Считается одним из самых простых типов заглушки. При передаче ей управления просто выводит сообщение о том, что это управление ею получено.
2. **Заглушка-константа.** Присваивает всем входным и вход- но-выходным параметрам некоторые постоянные значения.
3. **Заглушка-переменная.** Присваивает входным и входно- выходным параметрам допустимые спецификацией значения, различающиеся от

вызова к вызову. Может перебирать по циклу некоторый фиксированный набор значений или генерировать случайные значения.

4. **Заглушка-контролер.** Проверяет входные параметры на допустимое значение и выдает сообщение об ошибке или возвращает соответствующий код завершения.

Все виды заглушек можно комбинировать друг с другом.

На помощь вам могут прийти разнообразные варианты заглушек (mock objects). Есть несколько вариантов заглушек: mock, stub, dummy, spy, fake. Знать и различать их нужно и важно, поэтому стоит почитать и разобраться в специфике, но речь пойдет не об этом. Нам понадобится только один из видов заглушек, который наиболее подходит для работы с внешними сервисами — fake реализация.

Fake реализация представляет из себя упрощенный вариант настоящего сервиса, который работает полностью предсказуемо, локально, дает полный доступ к данным и настройкам, а также не требует никакой дополнительной работы в каждом тесте. Таким образом, мы сможем прозрачно заменить настоящий сервис в тестах на fake реализацию и решить все проблемы.

Во-первых, нам нужно будет реализовать этот упрощенный вариант сервиса, на что понадобится время и усилия разработчиков. Вы счастливчик, если ваш внешний сервис предоставляет fake реализацию для локального тестирования. Тогда вы просто скачиваете библиотеку, настраиваете, запускаете и пользуетесь на здоровье. Некоторые сервисы позволяют получить доступ к специальной версии внешнего сервиса для тестирования, которая не подвержена перечисленным в первой части проблемам. Но таких сервисов единицы. Почти всегда вам придется писать fake реализацию самостоятельно.

Во-вторых, вам придется позаботиться о том, чтобы ваша fake реализация ведет себя так же как реальный сервис. Это одни из самых частых граблей, на которые наступают сторонники описанного подхода. Вы делаете fake реализацию сегодня, а через неделю реальный сервис начинает вести себя по-другому. Ваши тесты по-прежнему проходят.

Эту проблему очень легко исправить, но придется приложить еще немного усилий. Вам нужно написать еще один набор тестов, теперь уже на реальный сервис, целью которого будет контроль и поддержание надежности протокола между вашим приложением и сторонним сервисом. Это некоторый вариант **smoke** тестов на внешний сервис, которые можно запускать раз в час, день или

неделю в зависимости от стабильности вашего внешнего сервиса. Тогда вы действительно контролируете и тестируете интеграцию с внешним сервисом.

Преимущества использования заглушки заключается в том, что она возвращает последовательные результаты, упрощая написание теста. Тесты можно выполнять, даже если другие компоненты пока не работают. Зачастую без них просто не обойтись, если тестируемая система с большим количеством зависимостей.

Недостатками заглушек является, то что они могут быть дорогостоящими (особенно это актуально для эмуляторов). Заглушки могут маскировать ошибки, могут устареть.